

PA 1085085

PCT / IB 03 / 05796
15 DEC 2003

REC'D 08 JAN 2004

WIPO

PCT

THE UNITED STATES OF AMERICA

TO ALL TO WHOM THESE PRESENTS SHALL COME:

UNITED STATES DEPARTMENT OF COMMERCE

United States Patent and Trademark Office

October 27, 2003

THIS IS TO CERTIFY THAT ANNEXED HERETO IS A TRUE COPY FROM THE RECORDS OF THE UNITED STATES PATENT AND TRADEMARK OFFICE OF THOSE PAPERS OF THE BELOW IDENTIFIED PATENT APPLICATION THAT MET THE REQUIREMENTS TO BE GRANTED A FILING DATE UNDER 35 USC 111.

APPLICATION NUMBER: 60/432,755

FILING DATE: December 12, 2002

PRIORITY DOCUMENT
SUBMITTED OR TRANSMITTED IN
COMPLIANCE WITH
RULE 17.1(a) OR (b)



By Authority of the
COMMISSIONER OF PATENTS AND TRADEMARKS

W. Montgomery
W. MONTGOMERY
Certifying Officer

12/12/02
13042 U.S. PTO

Please type a plus sign (+) inside this box →



12-13-02

60432755

PTO/SB/16 (02-01)

Approved for use through 10/31/2002. OMB 0651-0032
Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

PROVISIONAL APPLICATION FOR PATENT COVER SHEET

This is a request for filing a PROVISIONAL APPLICATION FOR PATENT under 37 CFR 1.53 (c).

Express Mail Label No.

INVENTOR(S)

Given Name (first and middle [if any])	Family Name or Surname	Residence (City and either State or Foreign Country)
Jan Jan-Willem	Hoogerbrugge van de Waerd	Eindhoven, The Netherlands San Jose, CA, USA

☐ Additional Inventors are being named on the _____ separately numbered sheets attached hereto

TITLE OF THE INVENTION (280 characters max)

COUNTER BASED STRIDE PREDICTION FOR DATA PREFETCH

CORRESPONDENCE ADDRESS

Direct all correspondence to:

☒ Customer Number

24738

OR

Type Customer Number here



24738

PATENT TRADEMARK OFFICE

☐ Firm or
Individual Name

Philips Intellectual Property and Standards

Address

1000 W. Maude Avenue

Address

City

Sunnyvale

State

CA

Zip

94085-2810

Country

USA

Telephone

408-617-7700

Fax

408-617-4856

ENCLOSED APPLICATION PARTS (check all that apply)

☒ Specification Number of Pages

13

☐ CD(s), Number

☒ Drawing(s) Number of Sheets

1

☐ Other (specify)

☐ Application Data Sheet. See 37 CFR 1.76

METHOD OF PAYMENT OF FILING FEES FOR THIS PROVISIONAL APPLICATION FOR PATENT (check one)

☐ Applicant claims small entity status. See 37 CFR 1.27.

☐ A check or money order is enclosed to cover the filing fees

FILING FEE
AMOUNT (\$)

☐ The Commissioner is hereby authorized to charge filing
fees or credit any overpayment to Deposit Account Number:

160

☒ Payment by credit card. Form PTO-2038 is attached.

The invention was made by an agency of the United States Government or under a contract with an agency of the United States Government.

☒ No.

☐ Yes, the name of the U.S. Government agency and the Government contract number are: _____

Respectfully submitted,

SIGNATURE

Robert M. McDermott

Date

12 December 2002

TYPED or PRINTED NAME

Robert M. McDermott

REGISTRATION NO.
(if appropriate)

41,508

Docket Number:

US02.0562P

TELEPHONE

804-493-0707

USE ONLY FOR FILING A PROVISIONAL APPLICATION FOR PATENT

This collection of information is required by 37 CFR 1.61. The information is used by the public to file (and by the PTO to process) a provisional application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 8 hours to complete, including gathering, preparing, and submitting the complete provisional application to the PTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, Washington, D.C., 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Box Provisional Application, Assistant Commissioner for Patents, Washington, D.C. 20231.

COUNTER BASED STRIDE PREDICTION FOR DATA PREFETCH

Technical Field

This invention relates to the field of electronics, and specifically to a method and system for predicting a next data location in a process to facilitate a prefetch of data from that location.

Background Art

Prefetching is a common technique for minimizing latency in a sequential process. Data that is expected to be needed at a future time in the process is retrieved from a memory and stored in a cache, for subsequent access at the future time. The cache is designed to provide a substantially faster access time than the memory. Thus, when the process needs the data, and the data is in the cache, the data is provided to the process at the higher cache-access speed. Conversely, if the data is not in the cache, the data is not provided to the process until after the substantially slower memory-access time, thereby introducing memory-access delays into the process.

A variety of techniques are commonly available to facilitate the prediction of the data that is going to be needed at a future time. One such technique is "stride prediction", wherein the location of the next data item to be needed is based on the sequence of locations of the prior-accessed data items. For example, data is often stored as an array or list of data records, such as records of employee's names, addresses, social security number, and so on. Typically, these records are fixed-length records, such that the beginning of each record is separated by a fixed number of memory locations from its adjacent records in the memory. An application that provides a printout or display of all employee names, for example, will sequentially access memory locations that are separated by this fixed number. Given that the application accesses a first employee name from the memory at location L , and accesses a second employee name at location $L+S$, it is likely that the application will access data located at location $L+S+S$ of the memory to obtain the next employee name. If the data at location $L+S+S$ is retrieved from memory and stored in a higher-speed cache memory before this access is requested, the third employee name can be provided to the application more quickly than the first or second employee names.

FIG. 1 illustrates an example flow diagram of a prior art stride prediction prefetch process. At 110, the prefetch process is invoked, typically at the same time that an

application invokes a request for access to a new data item. Not illustrated, this prefetch process may be called as part of a global scheme that includes multiple prefetching algorithms, and may be selectively invoked depending upon factors such as the proximity of sequential data accesses, and the like. Typically, a processor maintains a stride prediction table (SPT) that records information related to executed accesses to the memory, including the interval between sequential accesses, herein termed the stride of the accesses. The stride prediction table is typically configured to record multiple sets of information related to executed accesses to keep track of multiple potential strides. For ease of convenience and understanding, the invention is presented herein using the paradigm of a single set of information that is used to keep track of a single stride between related memory accesses. One of ordinary skill in the art will recognize that the principles presented herein are directly applicable to the conventional use of a stride prediction table that includes multiple sets of information related to multiple strides.

At 110, the current stride is determined by the difference between the address of the prior/old access and the address of the new requested access, at 120. On the assumption that a next requested access will be equally spaced as the prior access, a prefetch is executed to fetch the data at an address at the same interval from the current/new address, at 130. At 140, the new address replaces the old address, in preparation for the next data access, and the prefetch routine terminates, at 150. By initiating the prefetch for the next-likely data at the time of requested access to the new data, the prefetched data will be present in the higher-speed cache when and if the application initiates an access request for this data. If the next-likely data is not the data that is subsequently requested, the cache will not contain the next-requested data, and a memory-access will be required.

The prefetch process of FIG. 1, however, initiates a prefetch on every access to new data, without regard to whether there is any basis for the assumption that the next-likely requested data will be equally spaced from the prior-requested data. This causes substantial memory access traffic, which can serve to substantially reduce the effective memory access. FIG. 2 illustrates an example flow diagram of an improved prior art stride prediction prefetch process, wherein a prefetch is initiated if and only if two successive accesses exhibit the same stride. In this embodiment, at 220, the determined stride for the new access request is compared to the prior determined stride. If the new stride equals the

old stride, the likelihood that the next stride will also be equal to these two strides is sufficiently high to warrant a prefetch, at 130. If the new stride differs from the old stride, the new stride replaces the old stride, at 230, in preparation for the next cycle. One of ordinary skill in the art will recognize that this two-in-a-row criteria for initiating a prefetch may be extended to three-in-a-row, four-in-a-row, and so on, to tradeoff between excessive memory-access traffic and the likelihood of having the next-requested data in cache.

Description of the Invention

It is an object of this invention to improve the likelihood of prefetching data that will subsequently be accessed by an application. It is a further object of this invention to provide an efficient prefetch scheme that is well suited for hardware implementation.

These objects and others are achieved by a prefetching system that includes hysteresis in the determination and modification of a stride value that is used for prefetching data in a sequential process. Once a stride value is determined, intermittent stride inconsistencies are ignored, and the stride value retains its prior value. When the stride inconsistencies become frequent, the stride value is modified. When the modified stride value becomes repetitive, the system adopts this value as the stride, and subsequent stride inconsistencies are again ignored, and the stride value thereafter retains its current value until inconsistencies become frequent.

Brief Description of the Drawings

FIG. 1 illustrates an example flow diagram of a prior art stride prediction prefetch process.

FIG. 2 illustrates an example flow diagram of an alternative prior art stride prediction prefetch process.

FIG. 3 illustrates an example flow diagram of a stride prediction prefetch process in accordance with this invention.

FIG. 4 illustrates an example block diagram of a stride prediction prefetch system in accordance with this invention.

Throughout the drawings, the same reference numerals indicate similar or corresponding features or functions.

Best Modes for Carrying Out the Invention

This invention is premised on the observation that regular stride patterns in memory accesses often include intermittent data accesses that do not conform to the stride pattern. For example, a nested loop structure may include an inner loop that cycles through a list of data, and an outer loop that presets one or more variables that are used in each cycle of the inner loop, or an outer loop that stores a result from each cycle of the inner loop. In this example, the inner loop that is cycling through the list of data will likely exhibit a constant stride. At each re-commencement of the inner loop, however, the memory access is to the start of the list, whereas the prior access was to the end of the list. The span between the end of the list and the start of the list, however, will not correspond to the inner loop's stride. Additionally, a data access by the outer loop at the beginning or end of the loop will produce a span between accesses that does not correspond to the inner loop's stride. Other examples of an intermittent break in stride include the processing of data that is organized in a multi-dimension array. Typically, the data is processed for a given range along one dimension, then an index to another dimension is incremented, and the data at this next indexed other dimension is processed for the given range along the first dimension. The stride along the range of the first dimension will generally be constant, but the increment of the index to the next dimension will likely result in an access having a span from the prior access that does not match the stride along the first dimension.

In a conventional stride prediction process, whenever the stride is 'broken' or interrupted, the process of determining the stride is repeated to determine a new stride. During the time that the new stride is being determined, prefetches do not occur, and the application is delayed by memory-accesses at each re-start of an inner loop, or each incrementing of a higher-level index during the processing of a multi-dimension array.

In accordance with this invention, the stride value is preserved during intermittent breaks in stride. Performing a prefetch of data at the current stride is dependent upon the number of equal-value strides within a given number of memory accesses, and adjusting the prefetch value is dependent upon the occurrence of multiple non-equal-value strides. In a simple example, a prefetch of data may be performed whenever two out of three accesses have successive equal strides, and a modification of the stride value may be performed whenever two accesses have successive unequal strides.

FIG. 3 illustrates an example flow diagram of a stride prediction prefetch process in accordance with this invention. In this example, a "count" parameter is used to count the number of successive strides of the same value, up to a selected maximum. In a preferred embodiment of this invention, this count parameter is also used to distinguish between intermittent breaks in stride and an actual, persistent, change of stride. It will be evident to one of ordinary skill in the art in view of this disclosure that an independent parameter could be employed to count the number of successive non-equal strides.

After determining the current stride, between the new access address and the prior access address, at 120, the process of this invention compares the current stride with the prior stride, at 220. If the current stride equals the prior stride, the count is incremented, at 326; if not, the current count is decremented, at 322. In a preferred embodiment, the count is limited to a value from zero to a maximum count. Blocks 324 and 328 clip the incremented or decremented count from blocks 322 and 326 to remain within these limits, respectively.

At 330, the count is compared to an upper limit, UL, and to a lower limit, LL, wherein the lower limit LL is preferably less than the upper limit UL. In this example, the upper limit UL corresponds to the number of equal-stride occurrences required to warrant a prefetch from the next stride-predicted access location. If the count equals or exceeds this upper limit UL, a prefetch of data from an address corresponding to the current address plus the current stride is executed, at 130. For example, if the upper limit UL is a value of two, the prefetch is not executed, initially, unless two-in-a-row of the same stride value occurs. If the upper limit is a value of three, the prefetch is not executed until three-in-a-row of the same stride value occurs, and so on. Thereafter, subsequent equal-valued strides continue to increment the count, up to the maximum, via 326, 328. In a preferred embodiment, the maximum value is selected as the number of successive equal-value strides required to conclude that the current stride value is 'reliable'.

Each time that a non-equal stride occurs, the count is decremented, at 322. Thus, the difference between the maximum count and the current count corresponds to a measure of 'unreliability' of the current stride value. The lower limit is selected as the value of the current count that constitutes a sufficient measure of unreliability to warrant a change to the current stride value, at 230. Generally, as indicated in FIG. 3, if the current stride value

is deemed unreliable, at 230, a prefetch, at 130, is not performed. In like manner, if the current stride value is sufficiently reliable to warrant a prefetch, at 130, a modification of the current stride value at 230 is not performed. Thus, the lower limit LL is set to be less than the upper limit UL, and generally set to be $UL-1$. Optionally, as indicated by the
5 dashed connection between the test block 330 and the block 140, the lower limit LL may be selected relative to the upper limit UL such that the assessed reliability of the current stride is not sufficient to warrant a prefetch, at 130 ($count < UL$), while the assessed unreliability is not sufficient to warrant a modification of the current stride, at 230 ($count > LL$).

10 FIG. 4 illustrates an example block diagram of a stride prediction prefetch system 400 in accordance with this invention. A fetch controller 430 effects prefetching of data from a memory 450 to a cache 460, based on the contents of a control register 410 and the data access requests from a processor 420. The address of the requested data is used to determine whether the application is requesting data in a repetitive manner, exhibiting a
15 consistent stride, or span of locations, between memory accesses, as discussed above. The control register 410 includes an address of a prior data access 416, the prior stride 412, and a counter 414. As noted above, two counters may be used in lieu of the single counter 414, to maintain an independent count of equal-stride and non-equal stride accesses. The address of the currently requested data from the processor 420 is compared to prior address
20 416 to determine a current stride. If the current stride corresponds to the prior stride 412, the counter 414 is incremented; otherwise, it is decremented. Depending upon the incremented or decremented value of the counter 414, the fetch controller determines whether to initiate a prefetch of data from a next-predicted location in the memory 450 to the cache 460. When a subsequent data request is for data that has been prefetched into the
25 cache 460, the cache 460 provides the data directly to the processor, thereby avoiding the delay introduced by retrieving the data from the memory 450.

Also dependent upon the incremented or decremented value of the counter 414, the fetch controller 430 determines whether to modify the stride value 412, as detailed above. By determining whether to modify the stride value 412 based on a count that depends upon
30 the number of non-equal strides, rather than modifying the stride value based on the

occurrence of one non-equal stride as in a conventional system, the stride prediction prefetch system 400 is de-sensitized to intermittent breaks in stride.

The foregoing merely illustrates the principles of the invention. It will thus be appreciated that those skilled in the art will be able to devise various arrangements which, although not explicitly described or shown herein, embody the principles of the invention and are thus within the spirit and scope of the following claims.

CLAIMS

1. A method of prefetching data from a memory to a cache, comprising:
 - determining a first measure of equal-stride memory accesses, based on a plurality of equal-stride memory accesses and a prior stride value,
 - determining a second measure of non-equal-stride memory accesses, based on a plurality of non-equal-stride memory accesses and the prior stride value,
 - effecting a prefetch of the data from the memory based on the first measure, and
 - effecting a modification of the prior stride value based on the second measure.
2. The method of claim 1, wherein
 - determining the first measure and the second measure is effected by maintaining a count that is incremented for each equal-stride memory access and decremented for each non-equal-stride memory access, and
 - effecting the prefetch and effecting the modification are each based on the count.
3. The method of claim 2, wherein
 - effecting the prefetch occurs when the count is equal or above an upper limit, and
 - effecting the modification occurs when the count is equal or below a lower limit.
4. The method of claim 3, wherein
 - the count is limited to a maximum of three,
 - the upper limit is two, and
 - the lower limit is one.

5. A prefetch system comprising:

a control register that is configured to contain at least one measure that corresponds to a consistency of stride values between requested memory accesses,

a prefetch controller that is configured to prefetch data from a memory to a cache, based on the measure of consistency,

wherein

the consistency of stride values is dependent upon a comparison of a current stride with a prior stride value,

the prefetch controller is further configured to modify the prior stride value based on a measure of inconsistency, and

the measure of inconsistency is based on a plurality of non-equal-strides between requested memory accesses.

6. The prefetch system of claim 5, wherein

the measure of consistency and the measure of inconsistency correspond to a count that is incremented upon each equal-stride requested memory access, up to a maximum count, and decremented upon each non-equal-stride requested memory access, down to a minimum count.

7. The prefetch system of claim 6, wherein

the prefetch controller is configured to:

prefetch the data when the count is equal or above an upper threshold level, and

modify the prior stride value when the count is equal or below a lower threshold level.

8. The prefetch system of claim 7, wherein

the maximum count is three,

the upper threshold level is two,

the lower threshold level is one, and

the minimum count is zero.

9. A processing system, comprising:

a memory that is configured to provide access to data based on an access address,
a cache, operably coupled to the memory, that is configured to store data that is
accessed from the memory, to facilitate rapid access to the data,

a processor, operably coupled to the memory and the cache, that is configured to
provide the access address and to receive the data from the cache, if it is stored in the
cache, or from the memory, if it is not stored in the cache, and

a fetch controller, operably coupled to the processor, the memory, and the cache,
that is configured to effect a transfer of data from the memory to the cache, based on the
access address and a predicted stride value,

wherein

the fetch controller is further configured to maintain

a measure of stride consistency that is based on repeat occurrences of equal
stride values, and

a measure of stride inconsistency that is based on repeat occurrences of
unequal stride values, and

the fetch controller effects the transfer of data based on the measure of stride
consistency, and effects a modification of the predicted stride value based on the measure
of stride inconsistency.

10. The processing system of claim 9, wherein

the measure of stride consistency and the measure of stride inconsistency are each
based on a count that is incremented upon each equal-stride requested memory access, up
to a maximum count, and decremented upon each non-equal-stride requested memory
access, to a minimum count.

11

11. The processing system of claim 10, wherein

the fetch controller is configured to:

- effect the transfer of data when the count is equal or above an upper threshold level, and
- effect the modification of the predicted stride value when the count is equal or below a lower threshold level.

12. The processing system of claim 11, wherein

the maximum count is three,

the upper threshold level is two,

the lower threshold level is one, and

the minimum count is zero.

COUNTER BASED STRIDE PREDICTION FOR DATA PREFETCH

Abstract

5 A prefetching system (400) includes hysteresis in the determination and
modification of a stride value (412) that is used for prefetching data (130) in a sequential
process. Once a stride value is determined, intermittent stride inconsistencies are ignored
(322-330), and the stride value retains its prior value. When the stride inconsistencies
become frequent (322-330), the stride value is modified (230). When the modified stride
value becomes repetitive, the system adopts this value as the stride, and subsequent stride
10 inconsistencies are again ignored, and the stride value thereafter retains its current value
until inconsistencies become frequent.

13

1/2

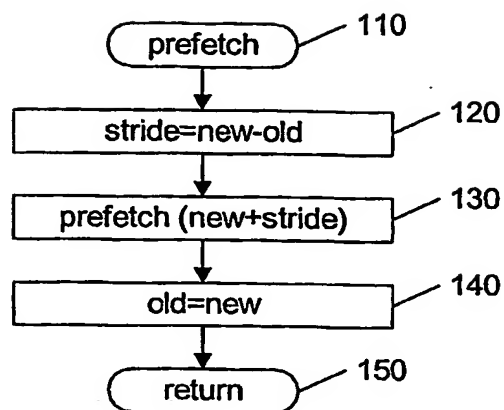


FIG. 1 [Prior Art]

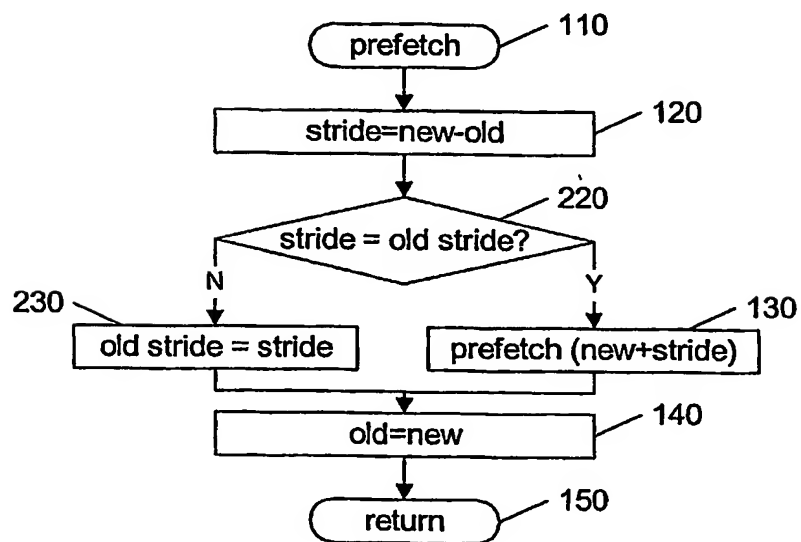


FIG. 2 [Prior Art]

14

2/2

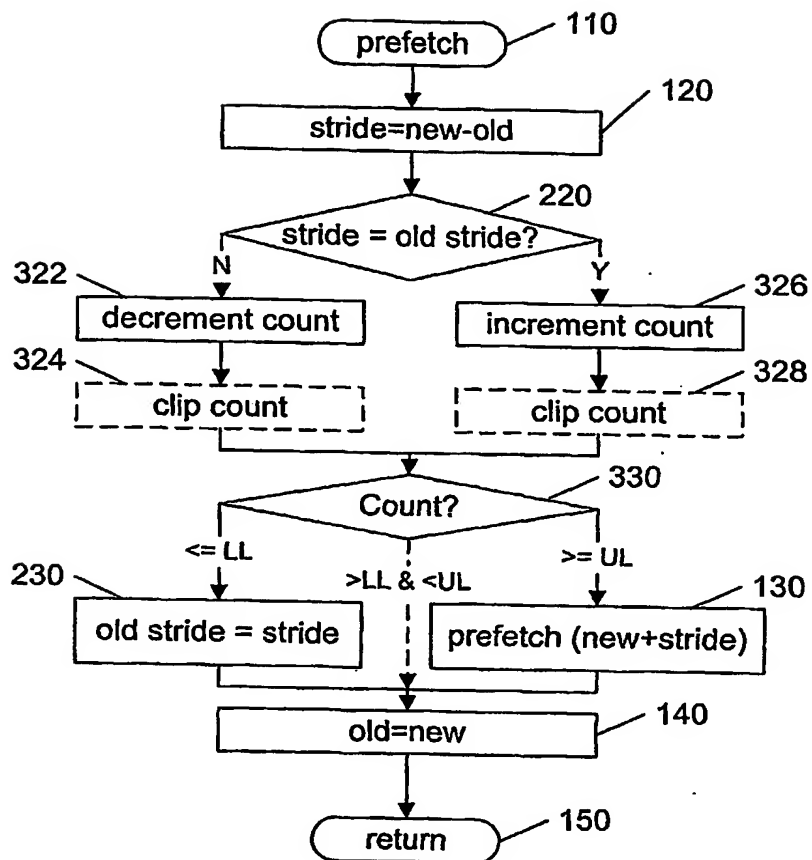


FIG. 3

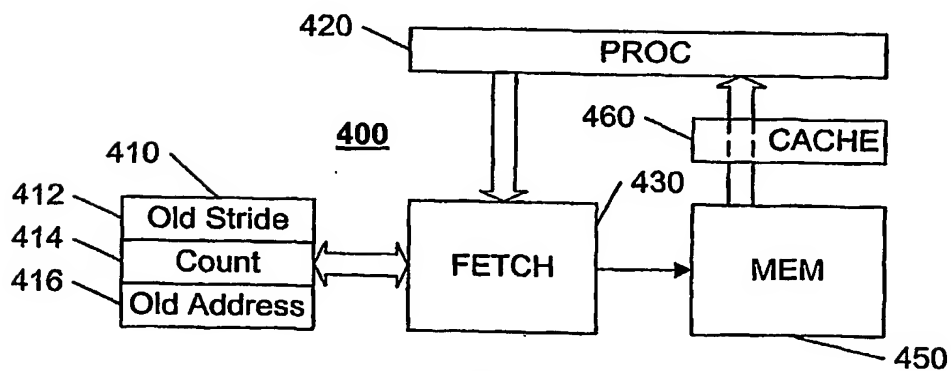


FIG. 4

**This Page is Inserted by IFW Indexing and Scanning
Operations and is not part of the Official Record**

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☒ BLACK BORDERS
- ☒ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☒ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☒ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.

As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.